객체-관계형 데이타베이스 시스템을 위한 새로운 성능 평가 방법론

김 성 진† · 이 상 호††

요 약

본 논문에서는 차세대 데이타베이스로 불리는 객체-관계형 데이타베이스 시스템에 대한 새로운 성능 평가 방법론(벤치마크)을 제안한다. 본 벤치마크는 객체-관계형 데이타베이스 시스템에 고유한 기능을 대상으로 성능 평가를 수행한다. 본 논문에서는 본 벤치마크의 설계 원칙, 시험 데이타베이스 구조, 시험 질의어 등을 기술한다. 본 벤치마크의 특징은 확장성, 합성 데이타베이스만의 사용, 질의 중심의 성능 평가 등이다. 우리는 본 벤치마크를 대표적인 두 개의 상용 객체-관계형 데이타베이스 시스템을 이용하여 구현하였으며, 그 성능 평가 결과도 보고한다.

A New Benchmark for Object-relational DBMSs

Sung-Jin Kim† · Sang-Ho Lee††

ABSTRACT

This paper presents a new benchmark for object-relational database systems, which are regarded as the next-generation database system. This benchmark has been developed to evaluate system performance peculiar to object-relational database systems. The design philosophy, test databases, and test queries of the benchmark are presented. This benchmark features scaleability, use of a synthesized database only, and a query-oriented evaluation. We have implemented this benchmark with two commercial object-relational database systems and the experimental results are also reported.

1. 서 론

1980년대 후반에 소개된 상용 객체-지향형 데이타베이스는 유연한 자료 모델을 제공한다. 그러나 객체-지향형 데이타베이스 시스템은 관계형 데이타베이스 시스템이 제공하는 유용한 기능(예를 들어, 동시성 제어, 회복 기능, 보안, 질의 언어와 최적화 등)을 충분히 제공하지 못하고 있다. 따라서 객체-지향형 데이타베이스는 CAD/CAM과 같은 특정 응용 분야에서 주로 사용이 되고 있다. 1990년대 초에 UniSQL과 Illustra사에 의해 소개된 객체-관계형(object-relational) 데이타베이스는 관계형 데이타 모델을 기반으로 객체-지향형 데이타 모델의 여러 기능을 지원한다.

데이타베이스 벤치마크는 여러 데이타베이스 시스템을 비교하여 각 시스템의 성능을 수치화한 것이다. 과거 20여년 동안 진행된 벤치마크의 주요 내용은 Wisconsin 벤치마크[2], TPC(Transaction Processing Council) 시리즈[4], AS^{3}AP[8], OO7 벤치마크[3], BUCKY[1] 등이 있다. 이들 벤치마크는 문제 영역 제시, 시험 질의 제안, 시험 결과 기술 등의 공통점을 가지나 문제 영역에 있어 다음과 같은 차이를 가진다. Wisconsin 벤치마크는 관계형 데이타베이스에 대한 기본적인 성능평가, OO7 벤치마크는 객체-지향형 데이타베이스에

※ 이 논문은 1998년 한국학술진흥재단의 학술연구비에 의하여 지원되었음.
† 준 회 원 : 숭실대학교 전산원 교수
†† 정 회 원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2000년 4월 4일, 심사완료 : 2000년 6월 21일

대한 성능 평가, TPC 시리즈 벤치마크는 트랜잭션에 대한 성능 평가를 문제 영역으로 다룬다. 특히, TPC 시리즈 벤치마크는 다중 사용자 환경에서 트랜잭션에 대한 성능 평가를 가능하도록 하였다. 데이타베이스 분야에서의 벤치마크에 관한 자세한 연구 및 개발 내용은 Gray[4]를 참조하기 바란다.

1996년 Wisconsin 대학에서 발표한 BUCKY는 객체-관계형 데이타 모델을 다루는 유일한 벤치마크로서 관계형 데이타 모델과 객체-관계형 데이타 모델의 차이에 대한 성능 평가를 목적으로 한다. 따라서 BUCKY는 객체-관계형 데이타 모델만을 위한 성능 평가로서 한계를 가진다. BUCKY는 동일한 시험 데이타베이스에 대해 객체-관계형 스키마(schema)와 관계형 스키마를 만든 후, 각 스키마에 동일한 기능의 객체-관계형 질의와 관계형 질의로써 성능 평가를 수행한다. BUCKY는 상속, 복합 객체 지원, 확장 자료형에 대한 상속, 객체간 참조(reference), 집합 속성, 객체간 메소드, ADT (abstract data type)와 메소드의 영역으로 나누어 성능 평가를 수행한다.

본 논문에서 우리는 객체-관계형 데이타베이스 시스템을 위한 새로운 성능 평가 방법론, 즉 벤치마크를 제안한다. BORD(Benchmark of Object-Relational Databases)라고 명명된 본 벤치마크는 객체-관계형 데이타베이스를 위한 새로운 성능 평가 방법론으로서 사용자 관점에서 유용한 객체-관계형 데이타베이스의 기능에 관하여 평가한다. 본 벤치마크는 확장성을 지원하며, 임의로 합성된 데이타베이스를 사용하며, 질의에 기반하는 데이타베이스 성능 평가 도구이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 벤치마크의 설계원칙, 시험 데이타베이스 구성 등을 기술한다. 3장에서는 시험 질의를 기술한다. 4장은 본 벤치마크의 구현 및 성능 평가 결과를 기술하며, 5장에서 결론을 맺는다.

2. 설계 전략 및 시험 데이타베이스

2.1 설계 전략

일반적인 관점에서 볼 때, 객체-관계형 데이타 모델은 관계형 데이타 모델 기초 위에 객체-지향적 데이타 모델의 유용한 기능(예를 들면, 상속, 데이타 추상화 등)을 수용하는 데이타 모델로 알려져 있다. 그러나 객체-관계형 데이타 모델에 관한 정확한 개념 및 기능에 관해서는 학계/산업계에서 아직까지 널리 수용되는 문서 또는 표준안이 없다. 객체 관계형 데이타베이스 시스템이 제공하여야 하는 정확한 기능은 아직 정립되지 않았으며, 또한 현재 판매되고 있는 상용 객체-관계형 시스템의 기능도 제품에 따라 많은 기능 차이를 보인다. 본 벤치마크는 Stonebraker[7]과 Kim[9]에 의거하여 객체-관계형 데이타베이스의 주요 기능을 항목별로 정립하였으며, 정립된 기능을 평가하는 성능 평가 방법론을 제시한다. 또한, 본 벤치마크는 특정한 응용 분야에 국한되지 않는 범용 분야에서 객체-관계형 데이타베이스 시스템의 성능 평가를 목표로 한다.

본 벤치마크는 객체-관계형 데이타베이스를 위한 성능 평가로 개발되었으며, 관계형 데이타베이스나 객체-지향 데이타베이스의 독특한 기능은 평가하지 않는다. 예를 들면, 객체-지향 데이타베이스에서만 사용되는 포인터 스위즐링(pointer swizzling)에 관한 성능은 본 벤치마크에서 평가하지 않는다. 객체-지향 또는 관계형 데이타 모델에 관한 독특한 기능은 기존의 성능 평가 방법론에서 이미 평가되고 있으므로 본 벤치마크에서 반복되지 않는다.

시험 데이타베이스는 실 데이타베이스를 사용하지 않고, 여러 조건을 만족하면서 인위적으로 생성된(synthesized) 데이타베이스를 사용한다. 실 데이타베이스는 실세계의 특정 분야 데이타 특성을 반영한다는 장점을 가지고 있으나 시험 데이타베이스로서 다음과 같은 두 가지 단점을 가진다. 첫째, 시험 데이타베이스의 확장성이 전혀 없다. 둘째, 데이타 분포의 무작위성으로 인하여 시험 결과에 대한 검증 작업이 실질적으로 불가능하다.

본 벤치마크에서 사용되는 시험 데이타베이스는 데이타 분포 관점에서 순차 분포(sequential distribution), 원형(circular) 분포, 무작위(random) 분포를 갖는다. 순차 분포 데이타의 값은 해당 인스턴스만큼 증가하므로 모든 값이 유일(unique)하다. 원형 분포 데이타의 값은 일정한 간격을 두고 반복되어 중복된 데이타들이 생성되지만, 같은 값의 데이타 개수를 조절할 수 있다. 무작위 분포 데이타는 시험 질의에서는 사용되지 않으며, 다른 용도(즉, 데이타베이스 인스턴스를 무작위로 분포시킬 때)로만 사용된다.

시험 데이타베이스의 크기는 DBMS의 성능에 지대한 영향을 미친다. 본 벤치마크는 확장 요소(scale factor)를 통해 사용자가 시험 데이타베이스 크기를 조절

할 수 있도록 한다. 사용자는 사용하고자 하는 환경에 적합한 데이타베이스 크기를 성능 평가 시에 반영할 수 있으므로, 현실성 있는 데이타베이스 시스템 성능 평가 결과를 얻을 수 있다.

인덱스 구조와 인덱스 생성이 가능한 자료형은 데이타베이스 시스템에 따라 차이가 있다. 본 벤치마크는 인덱스 생성에 있어서 제한을 두지 않으며, 사용자가 가능한 많은 인덱스를 생성하는 것을 허용한다. 이는 다양한 인덱스를 지원하는 시스템에게 보다 좋은 평가(evaluation, credit)를 주기 위함이다.

본 벤치마크의 사양은 각 DBMS의 문법이나 기능상의 차이에 의해 다르게 구현이 될 수 있으나, 기본적으로 사용자나 데이타베이스 프로그래머가 응용 프로그램에서 DBMS가 제공하지 않은 기능을 구현하는 것을 허용하지 않는다. 즉, DBMS에서 제공하는 기능만을 통하여 평가를 수행한다. 이는 많은 기능성을 제공하여 좋은 성능을 나타나도록 해주는 시스템에게 보다 많은 평가를 주기 위함이다.

본 벤치마크에서 성능 평가의 측정은 제안된 각 질의에 대해 단일 사용자 환경에서 시스템의 응답 시간으로 한다. 응답 시간의 측정은 10번 콜드 런(cold run)을 반복하여 산술 평균으로 한다. 콜드 런이란 질의 반복 수행 시 버퍼(buffer)가 비워져 다음 질의의 수행에 영향을 미치지 않는 것을 말한다. 본 벤치마크는 응답 시간 외의 평가 항목으로 QPM(queries per minute)과 P/QPM(price per QPM)을 고려한다. QPM과 P/QPM은 분당 수행할 수 있는 질의 수와 분당 수행할 수 있는 질의 수에 대한 비용을 뜻한다. QPM은 성능 평가의 결과를 단일 수치로 하여 결과의 이해와 각 시스템과의 비교를 용이하게 한다. 이때, 대형 컴퓨터와 개인용 컴퓨터는 성능과 가격에 있어서 일장일단을 가진다. 예를 들어, 대형 컴퓨터는 개인용 컴퓨터에 대해 우수한 성능을 보이지만 훨씬 많은 비용을 필요로 한다. P/QPM은 비용을 고려하여 QPM 결과를 정규화하는 것으로, 여기서 비용이라 함은 5년 동안의 하드웨어와 소프트웨어에 소요된 모든 비용을 뜻한다. P/QPM은 사용자에게 소형에서부터 대형까지 다양한 컴퓨터 환경에서 본 성능 평가 결과를 활용할 수 있도록 한다.

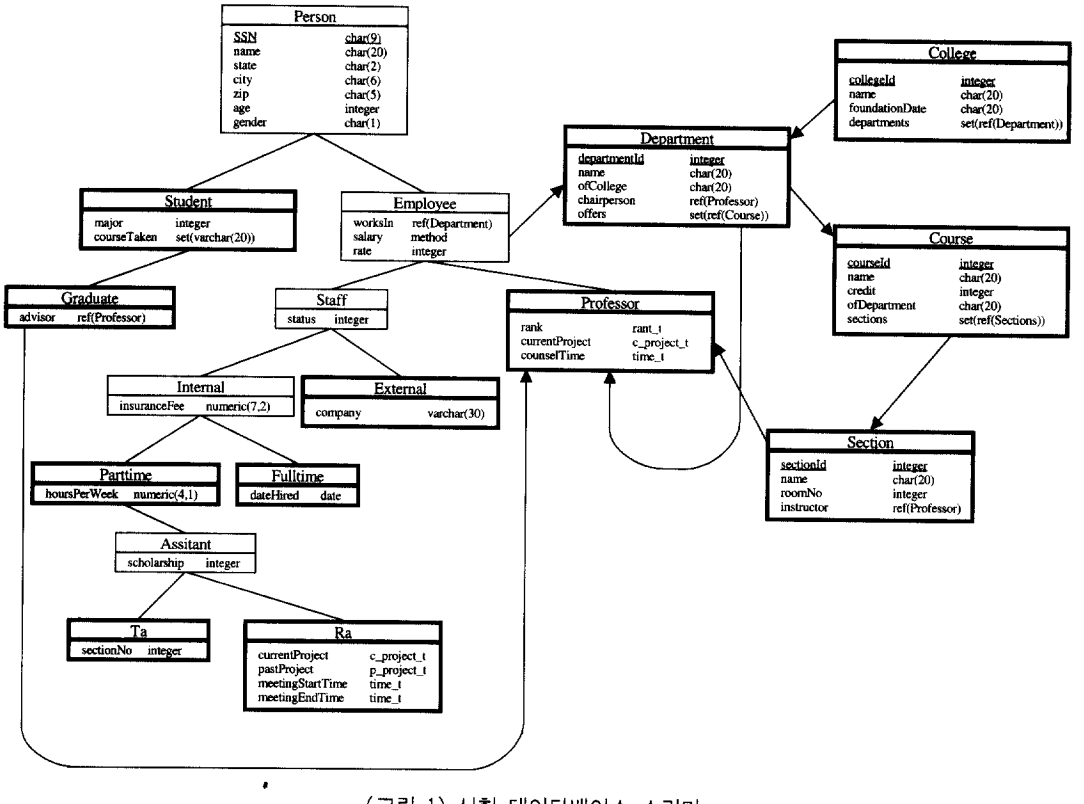
2.2 시험 데이타베이스 스키마와 인스턴스

시험 데이타베이스의 스키마와 인스턴스는 (그림 1)과 같은 일반적인 대학교 환경을 모델링한다. (그림 1)에서 일반적인 직사각형은 인스턴스가 없는 클래스를 나타내고, 굵은 선의 직사각형은 인스턴스가 있는 클래스를 나타낸다. 클래스를 나타내는 직사각형은 클래스 이름, 클래스내의 속성, 각 속성의 자료형으로 표현된다. 각 클래스에서 밑줄이 있는 속성은 키 속성을 의미한다. 일반적인 직선은 상속 관계(inheritance relationship)를 나타내며, 방향성을 갖는 직선은 참조 관계(reference relationship)를 나타낸다. 본 벤치마크의 시험 데이타베이스에는 총 17개의 클래스가 있으며, 상속성의 깊이는 7이고, 4개의 홉(hop)을 가진다.

주요한 인스턴스의 분포는 다음과 같다. Person 클래스의 SSN 속성은 주 키 속성으로 중복되지 않는 유일 값을 가진다. SSN 속성은 9자리의 문자열로서 앞의 두 자리 문자열은 해당 클래스를 나타내며 나머지 7자리 문자열은 클래스 내의 유일한 값으로 할당된다. 클래스의 name 속성은 스키마 내의 모든 클래스가 가지고 있는 속성으로 무작위 값을 갖는 20자리 문자열로 이루어진다. city, state 속성은 각각 1%와 10%의 키 평균 분포(key average distribution)를 갖는다. 예를 들어, city 속성값은 서로 다른 100개의 값이 균등분포(uniform distribution)되어 있어, 특정 하나의 값을 조건으로 하여 정확 매치(exact match) 질의를 할 경우, 정확하게 1% 인스턴스가 질의 조건을 만족한다. age 속성은 속성값으로 20에서 60까지의 값을 가지나 인스턴스의 1%는 60의 값을 갖도록 한다. zip 속성은 0부터 9999까지의 순차값을 갖는다. gender 속성은 속성값으로 'M'과 'F'를 가지며, 50%의 키 평균 분포를 보인다.

Student 클래스의 courseTaken 속성은 1-20개의 원소 개수를 가지는 집합형 데이타 형태이며, 원소 하나의 크기는 5바이트이다. Graduate 클래스의 courseTaken 속성은 201-250개의 원소 개수를 가지는 집합형 데이타 형태이며, 원소 하나의 크기는 5바이트이다. 집합형 속성내의 원소값 분포 또한 질의 조건절에서 선택율을 조절할 수 있도록 생성된다.

Student 클래스의 major 속성은 Department 클래스의 departmentId 속성을 참조하기 위한 외래키이다. Generic ADT 시험을 위한 자료형으로 Professor 클래스에 rank_t, c_project_t 자료형, Ra 클래스에는 time_t, c_project_t, p_project_t 자료형이 있다. ADT와 관련한 설명은 3장에서 더 자세히 다루어질 것이다.



(그림 1) 시험 데이타베이스 스키마

본 벤치마크는 확장 요소를 통해 벤치마크의 확장성을 제공한다. 확장 요소(scale factor, SF)는 사용자가 시험 데이타베이스를 필요에 따라 적당한 크기로 생성할 수 있도록 한다. 기본적인 데이타베이스 크기는 <표 1>과 같으며 사용자는 확장 요소를 통하여 인스턴스의 개수를 조절할 수 있다. 확장 요소로는 소수를 허락하지 않으며 자연수만을 허락한다.

〈표 1〉 시험 데이타베이스 인스턴스

Class	*Instance*	*Class*	*Instance*
Student	100,000 * SF	Professor	30,000 * SF
Graduate	60,000 * SF	External	40,000 * SF
Department	500 * SF	Fulltime	20,000 * SF
College	100 * SF	Parttime	10,000 * SF
Section	60,000 * SF	Ta	50,000 * SF
Course	30,000 * SF	Ra	50,000 * SF

3. 시험질의

성능 평가를 위한 질의는 그 기능에 따라 9개 항목으로 분류되며, 총 28개의 질의를 시험한다. 참고로 현재까지 객체-관계형 데이타베이스를 위한 표준화된 SQL 언어는 부재하다. 각 질의는 서술식 기술(description)과 실제 상용 데이타베이스 시스템에서 사용되는 SQL을 기반으로 하여 표현한다.

3.1 클래스(Class)

클래스는 ORDBMS에서 기본적인 모델링 단위이다. 상속성은 ORDBMS에서 제공하는 주요한 특징중 하나이다. Q1-1은 하나의 클래스에 대한 기본적인 질의로서 객체-관계형 모델의 전형적 기능은 아니지만 전체 성능 평가에 대한 기준이 된다. Q1-2는 계층구조에 대한 질의로서 Q1-1과 같은 결과를 반환한다. 이는 계층구조에 대한 인덱스 사용의 효율성을 시험한다.

Q1-1 단일 클래스 정확 매치 : 대학원생 중에 주민번호가 120000050인 사람의 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

Select name, state, city, zip, age, gender from

only(Graduate)
where SSN = '120000050';

Q1-2 클래스 계층 정확 매치 : 모든 사람들 중에 주민번호가 120000050인 사람의 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

Select name, state, city, zip, age, gender from Person
where SSN = '120000050';

3.2 객체 아이디(OID)

ORDBMS에서는 OODBMS와 마찬가지로 클래스나 인스턴스에 대하여 식별할 수 있는 고유한 OID(Object ID)기능을 제공한다. 각 개체들의 OID는 객체-관계형 데이타베이스 시스템의 필수적인 기능이다. Q2-1과 Q2-2는 하나의 OID를 추출하는 질의와 OID의 집합을 추출하는 질의 성능을 시험한다.

Q2-1 단일 OID 추출 : 번호가 10번인 학과의 학과장 OID를 검색한다.

Select chairperson from Department where departmentId = 10;

Q2-2 OID 집합 추출 : 번호가 10번인 학과에서 제공하는 과목들의 OID를 검색한다.

Select offers from Department
where departmentId = 10;

3.3 조인(Join)

ORDBMS에서의 조인은 기존의 RDBMS에서의 특성과 OODBMS에서의 특성을 모두 가지고 있다. 기존 RDBMS의 조인과 차이점은 클래스 계층 구조에 대한 조인 여부이다. Q3-1, Q3-2, Q3-3은 선택율에 따른 질의를 시험한다.

Q3-1 검색 조건 없는 클래스 계층 조인 : 각 학생의 학생번호와 학과의 이름을 검색한다.

Select SSN, d.name from Department d, Student
where departmentId = major;

Q3-2 검색 조건 있는 클래스 계층 조인(1% 선택율) : city25 도시에 사는 학생의 학생번호와 학과의 이름을 검색한다.

Select SSN, d.name from Department d, Student
where departmentId = major and city = 'city25';

Q3-3 검색 조건 있는 클래스 계층 조인(10% 선택율) : S5 도에 사는 학생의 학생번호와 학과의 이름을 검색한다.

Select SSN, d.name from Department d, Student
where departmentId = major and state = 'S5';

3.4 참조(Reference)

ORDBMS는 다른 클래스를 참조할 수 있는 도메인(domain)을 속성으로 가질 수 있다. 이 경우 해당 인스턴스를 찾기 위하여 OID를 사용한다. 시험 데이타베이스에서 구성된 참조 관계는 2-홉(2-hop)의 참조 관계를 시험할 수 있는 구조를 가지고 있다. Q4-1, Q4-2. Q4-3, Q4-4는 1,2 단계의 참조 관계에 대하여 각기 1%, 10%의 선택율을 가지도록 질의를 작성하여 성능을 시험한다.

Q4-1 1-홉 참조(1% 선택율) : city50 도시에 사는 Ta의 주민번호와 일하고 있는 학과의 이름을 검색한다.

Select SSN, worksIn->name from Ta
where city = 'city50';

Q4-2 1-홉 참조(10% 선택율) : S5 도에 사는 Ta의 주민번호와 일하고 있는 학과의 이름을 검색한다.

Select SSN, worksIn->name from Ta
where state = 'S5';

Q4-3 2-홉 참조(1% 선택율) : city50 도시에 사는 Ta의 주민번호와 일하고 있는 학과의 학과장 이름을 검색한다.

Select SSN, worksIn->chairperson->name from Ta where city = 'city50';

Q4-4 2-홉 참조(10% 선택율) : S5 도에 사는 Ta의 주민번호와 일하고 있는 학과의 학과장 이름을 검색한다.

Select SSN, worksIn->chairperson->name from Ta
where state = 'S5';

3.5 집합(Set)

ORDBMS는 RDBMS에서 제공하지 않는 집합 자료형을 가지고 있다. 집합 자료형은 집합(set), 다중집합(multiset), 리스트(list)의 세 가지로 분류할 수 있으며, 본 벤치마크는 원소들간의 중복을 허용하지 않는 집합

형을 시험한다. 집합은 크기에 따라 소 집합과 대 집합으로 나뉜다. 소 집합은 최소 1부터 최대 20개의 원소 개수를 갖는 집합이며, 대 집합은 최소 201개부터 최대 250개의 원소를 갖는 집합이다.

DBMS에서 집합 자료형을 저장 방법은 크게 두 가지로 나뉜다. 인-라인(in-line) 집합[7]으로 불려지는 첫 번째 저장 방식은 집합 원소들을 물리적으로 근접한 공간상에서 다른 데이타 속성과 함께 저장하는 것이다. 이러한 방식은 작은 원소의 개수를 가진 집합 저장에 유리하다. 두 번째 방식은 집합 원소들을 독립된 파일로 저장하여, 집합 원소에 대한 직접적인 접근을 허용하지 않으며, 원소들에 대하여 참조 무결성을 이용하여 2단계 접근 방식을 제공하는 방식이다. 이러한 방식은 집합의 원소들이 많은 경우 또는 원소들의 입력, 변경, 삭제 등이 빈번한 경우에 적합하다.

집합 시험 질의로 총 4개를 정의한다. 이중 Q5-2와 Q5-4는 50%의 선택율을 갖는 질의로서 직렬 집합으로 구현된 시스템에 보다 나은 평가를 주도록 설계되었다. Q5-1과 Q5-3은 10%의 선택율을 갖는 질의로서 집합자료형에 대한 인덱스의 구현이 가능한 시스템에 보다 나은 평가를 주기 위하여 설계되었다.

Q5-1 소 집합(10% 선택율) : 과목 'saaaa'를 수강한 학부 학생들만의 주민번호를 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from only(Student)
where courseTaken ⊇ {'saaaa'};
```

Q5-2 소 집합(50% 선택율) : 과목 'gaaaa', 'haaaa', 'iaaaa', 'jaaaa', 'kaaaa'를 수강한 학부 학생들만의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from only(Student)
where courseTaken⊇{'gaaaa','haaaa',...,'kaaaa'};
```

Q5-3 대 집합(10% 선택율) : 과목 'febaa'를 수강한 대학원생들만의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from only(Graduate)
where courseTaken ⊇ {'febaa'}
```

Q5-4 대 집합(50% 선택율) : 과목 'gbcaa', 'hbcaa', 'ibcaa', 'jbcaa', 'accaa', 'bccaa', 'cccaa', 'dccaa', 'eccaa', 'fccaa'를 수강한 대학원생들만의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from only(Graduate)
where courseTaken ⊇
{'gbcaa', 'hbcaa', ..., 'fccaa'};
```

3.6 메소드(Method)

ORDBMS에서 지원하는 메소드는 크게 DBMS 자체적으로 제공하는 내장 함수와 사용자가 직접 정의하여 사용하는 사용자 정의 메소드로 구분한다. 본 벤치마크는 관계형 데이타베이스에서 지원하지 않는 사용자 정의 메소드를 시험한다. 우선, 본 벤치마크는 고용인의 봉급을 계산하는 salary 메소드를 정의한다. Employee 클래스에 대해 정의된 salary 메소드는 모든 하위 클래스에서 상속을 받게 되며, 함수식은 아래와 같다.

$$salary = rate * SALARYCONSTANT$$

상기식에 사용되는 변수를 살펴보면 rate는 Employee 클래스에서 정의된 속성이고, SALARYCONSTANT는 상수이다. 이때 Ta 클래스는 상속받은 salary 메소드를 사용하지 않고, 클래스 자체에서 정의한 salary 메소드를 사용한다.

$$salary = (basesalary * (age / AGELEVEL))^{BASESALARYRATE} + ceil((SALARYCONSTANTTA * hoursPerweek) * rate) - floor(\sqrt{insuranceFee + basesalary * INSURANCERATE})$$

상기식은 Ta 클래스에서 사용되는 salary 메소드이다. Ta 클래스는 Employee 클래스의 salary 메소드를 상속받지만 salary 메소드에 대해 오버로딩(overloading)함으로써 자신이 생성시킨 고유의 메소드를 사용한다. 사용된 변수를 살펴보면 대문자로 시작하는 변수들은 상수이다. 또한, basesalary 변수를 제외한 소문자로 시작하는 변수들은 Ta 클래스가 가지고 있는 속성이다. basesalary는 기본급으로서 나이에 따라 200,000에서 300,000 사이의 수를 할당한다. Q6-1, Q6-2, Q6-3은 메소드에 대한 비교 연산, 추출, 질의 최적기를 평가한다.

Q6-1 단일 클래스 메소드 : 성별이 남자이고 월급이 2500000이 넘은 Ta들의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age from
only(Ta)
where gender = 'M' and salary(Ta) >= 2500000;
```

Q6-2 "Select" 구문에서의 단일 클래스 메소드 : 성별이 남자인 Ta들만의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별, 월급을 검색한다.

```
Select SSN, name, state, city, zip, age, salary(Ta)
from only(Ta)
where gender = 'M';
```

Q6-3 클래스 계층 메소드 : 월급이 2500000이 넘고, 성별이 남자인 모든 직원들의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from Employee
where salary(Employee) >= 2500000 and gender =
'M';
```

3.7 연산자(Operator)

객체-관계형 데이타베이스는 사용자 정의 연산자를 지원해야 한다[7]. 또한, 객체-관계형 데이타베이스는 사용자 정의 연산자의 특성을 고려한 질의 실행 계획이 가능하여야 한다. 본 벤치마크에서는 사용자 연산자로서 '<<'와 '>>'를 정의한다. '<<'는 오른쪽 피연산자가 왼쪽 피연산자보다 클 때 '참' 값을 반환하고, '>>'는 왼쪽 피연산자가 오른쪽 피연산자보다 클 때 '참' 값을 반환하는 연산자이다. Q7-1, Q7-2, Q7-3은 내장 연산자, 사용자 정의 연산자, 사용자 정의 연산자에 대한 질의 최적기를 시험한다. 또한 1%의 선택율로 시험하여 사용자 정의 연산자에 대한 인덱스 지원이 가능한 시스템에 보다 나은 평가를 줄 수 있도록 설계되었다.

Q7-1~Q7-3(1% 선택율) : 나이가 60 이상인 Ta들만의 주민번호, 이름, 도, 도시, 우편번호, 성별을 검색한다.

Q7-1 : 내장 연산자

```
Select SSN, name, state, city, zip, gender from
only(Ta) where age >= 60;
```

Q7-2 : 사용자 정의 연산자

```
Select SSN, name, state, city, zip, gender from
only(Ta) where age >> 60;
```

Q7-3 : Q7-2의 교환 연산자

```
Select SSN, name, state, city, zip, gender from
only(Ta) where 60 << age;
```

3.8 플래트닝(Flattening)

ORDBMS에서의 메소드는 프로그램 언어로 쓰여지거나 SQL로 쓰여질 수 있다. 메소드가 SQL로 작성된 경우에는 질의 최적화 단계에서 메소드를 포함한 전역적인 최적화가 가능하나[7], 메소드가 C 언어와 같은 프로그램 언어로만 작성이 가능한 경우는 전역적인 질의 최적화를 할 수 없다.

Q8-1은 플래트닝이 가능한 시스템에 보다 나은 평가를 주도록 설계되었다. Q8-1에 사용된 malegraduate()는 SQL내에서 작성된 것으로서 Graduate 클래스에서 모든 남자 대학원생을 반환하는 메소드이다. malegraduate()를 SQL로 표현하면 아래와 같다.

```
Select * from only(Graduate) where gender = 'M';
```

만약 전역적인 질의 최적화가 가능하다면, SSN 속성에 대한 인덱스 검색을 우선하여 Q1-1의 결과와 비슷한 응답시간이 소요될 것이다.

Q8-1 단일 클래스에 대한 복잡 질의 플래트닝 : 성별이 남자이고 주민번호가 120000050인 사람의 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select name, state, city, zip, age, gender from
malegraduate()
where SSN = '120000050';
```

3.9 일반(Generic) ADT

DBMS에서 제공하는 자료형은 내장 자료형(예를 들어, integer, char, date 등)과 ADT 자료형이 있다. 내장 자료형은 DBMS에서 기본적으로 제공하는 자료형이다. DBMS는 내장 자료형에 대해 내부적 저장 구조를 결정하고 이에 따른 연산도 같이 제공한다. ADT 자료형은 사용자에 의해 내부 구조가 정의된 자료형이다. DBMS는 ADT 자료형의 내부 구조를 이해하지 못하므로 사용자는 ADT 자료형에 접근할 수 있는 연산자를 제공하여야 한다.

ADT 자료형은 단일 자료형(atomic type) ADT와 복합 자료형(composite type) ADT로 구분된다. 단일

자료형 ADT는 단일 값(atomic value)을 갖는 자료형이다. 예를 들어, 체중 속성이 weight_t 라는 단일 자료형 ADT로 정의되면, 하나의 인스턴스가 체중의 속성값으로 가지는 값은 오직 하나이다. 복합 자료형 ADT는 하나 이상의 값을 가질 수 있는 자료형이다. 예를 들어, 주소 속성이 address_t 자료형으로 정의되고 address_t가 nation, state, street, zip의 4개 항목으로 구성된 복합 자료형 ADT이면, 하나의 인스턴스는 주소의 속성값으로서 '주소.nation', '주소.state', '주소.street', '주소.zip'을 위한 4개의 값을 가진다. address_t 자료형은 마치 C/C++언어에서 struct와 같이 선언된다. address_t 자료형으로 정의된 속성은 각 항목의 구분을 위해 태그(tag)를 필요로 한다.

시험 데이타베이스 스키마에서 복합 자료형 ADT로는 c_project_t 자료형과 p_project_t 자료형이 있다. c_project_t 자료형은 4개의 항목으로 구성된다. 각 항목의 이름은 projectId, projectName, projectStartDate, projectEndDate이고 항목의 자료형은 내장 자료형인 integer, char(20), date, date로 정의된다. p_project_t 자료형은 일종의 집합 자료형으로 같은 자료형의 항목을 여러 개 가진다. 단일 자료형 ADT로는 rank_t 자료형과 time_t 자료형이 있다. Professor 클래스의 rank_t 자료형은 내장 자료형의 char(5)와 같다. time_t 자료형은 시간 데이타를 저장하기 위해 정의된 자료형이다. time_t 자료형은 시간과 분에 대한 정보를 포함하며 0시 0분부터 23시 59분까지의 범위를 저장할 수 있다. 시간과 분의 정보는 ':' 문자로 구분된다. DBMS는 time_t에 대한 내부 구조를 알지 못하므로 이를 다루기 위해 I/O 연산자와 여섯 가지의 비교 연산자(즉, =, ≠, 〈, 〉, ≥, ≤)가 정의되어야 한다.

본 벤치마크는 ADT 시험을 위한 3가지 카테고리를 제안한다. 카테고리 9는 ADT에 대한 단순 조건 질의, 카테고리 10은 ADT에 대한 단순 연산 질의, 카테고리 11은 ADT에 대한 조인 질의이다. 각각의 카테고리에는 단일 자료형 ADT와 복합 자료형 ADT에 대한 시험을 포함한다.

Q9-1 단일 자료형 ADT 접근(1% 선택율) : 회의의 종료 시간이 오후 2시 30분인 Ra의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from Ra
where meetingEndTime = '14 : 30';
```

Q9-2 복합 자료형 ADT 접근 (1% 선택율) : 현재 진행중인 프로젝트의 번호가 10인 교수의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from Professor
where currentProject.projectId = 10;
```

Q9-2에서 projectId는 복합 자료형 ADT의 첫 번째 필드이름이다. Q10-1과 Q10-2에서 사용되는 elapsedTime() 연산자는 time_t 자료형의 두 인수를 받아 주어진 시간의 차이를 반환한다. Q10-2에서의 projectTerm() 연산자는 c_project_t 자료형의 인수를 받아 c_project_t 자료형의 하위 projectStartDate, projectEndDate 필드 값의 차이를 정수형으로 반환한다.

Q10-1 단일 자료형 ADT에 대한 연산 : 회의 시간이 1시간 이하인 Ra의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from Ra
where elapsedTime(meetingStartTime,
meetingEndTime) <= 1;
```

Q10-2 복합 자료형 ADT에 대한 연산 : 1년 이하의 프로젝트를 진행중인 교수의 주민번호, 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
Select SSN, name, state, city, zip, age, gender
from Professor
where projectTerm(currentProject) <= 365;
```

Q11-1 단일 자료형 ADT에 대한 조인 : 교수의 상담시간과 Ra의 회의 끝 시간이 같은 교수와 Ra의 이름을 각각 검색한다.

```
Select p.name, r.name from Professor p, Ra r
where p.counselTime = r.meetingEndTime;
```

Q11-2 복합 자료형 ADT에 대한 조인 : 교수의 현재 프로젝트 번호와 Ra의 현재 프로젝트 번호가 같은 교수와 Ra의 이름을 각각 검색한다.

```
Select p.name, r.name from Professor p, Ra r
where p.currentProject.projectId =
r.currentProject.projectId;
```

4. 구현 및 결과

우리는 본 벤치마크를 두 개의 상용 DBMS 제품으로 구현하였다. 구현 목적은 상용 시스템의 성능 비교가 아니라, 본 벤치마크의 가시성(feasibility)을 위한 것이다. 시험에 사용된 DBMS의 제품명은 여러 이유로 인하여 명시하기가 곤란하므로, 본 논문에서는 각각 시스템을 X와 Y로서 표기하기로 한다.

시험 환경은 썬 울트라 스팍(Sun Ultra Sparc) 기종이며, 96M의 주 메모리가 장착되어 있다. 시스템에 따라 필요한 경우, 32M의 공유 메모리가 사용되었다. 운영 체제는 솔라리스(Solaris) 2.5.1이 사용되었고, 시험을 위한 응용 프로그램은 ESQL/C로 구현되었다. 시험에 사용되는 데이타베이스는 원시 디스크(raw disk)를 사용하지 않고 유닉스(Unix)의 파일시스템을 이용하여 저장하였다.

본 벤치마크의 사양을 지원하는 정도에 따라 X와 Y 시스템을 위한 스키마가 생성되었다. 예를 들어 X는 클래스간의 참조를 지원하지 않으나 Y는 이를 지원한다. 본 벤치마크가 제시한 스키마에 의하면 Department 클래스, Course 클래스, Section 클래스, Professor 클래스는 환형 참조(circular reference)의 관계를 가진다. 환형 참조 관계에 있는 클래스들은 자신의 클래스가 생성되기 이전에 다른 클래스가 먼저 생성되기를 기다리므로 어떠한 클래스도 먼저 생성될 수 없는 문제점을 가지고 있다. 우리는 환형 참조의 문제점을 해결하기 위해 우선 Department 클래스에 chairperson, offers 속성(이들은 각기 Professor와 Course 클래스를 참조한다)이 제외된 Department 클래스를 생성하였다. Department 클래스의 생성이후 각 클래스들은 Professor 클래스, Section 클래스, Course 클래스의 순서로 생성되었고 마지막으로 "alter" 구문을 사용하여 Department 클래스에 chairperson, offers 속성을 추가하였다.

스키마에 따른 인스턴스 생성은 X와 Y에서 제공하는 벌크 로딩(bulk loading) 기능을 이용하였다. 우선, 우리는 각 클래스들의 데이타를 아스키 파일로 생성하였다. 아스키 파일내의 데이타는 일정한 형식을 가지게 되는데 각 인스턴스의 구분을 위해 캐리지 리턴(CR, carriage return)이 사용되었고 속성값의 구분을 위해 X와 Y에서 제공하는 각각의 구분자가 사용되었다. 생성된 데이타는 주 키 속성에 대해 오름차순으로 정렬되어 있다. 또한 이들 데이타들은 본 벤치마크에서 제시하는 다수의 제약조건을 모두 만족시키도록 생성되었다. 예를 들어 Section 클래스, Course 클래스, Department 클래스, College 클래스는 서로 참조 관계가 있으며 참조되는 클래스의 인스턴스 개수가 참조하는 클래스의 인스턴스 개수에 배수가 된다. 다른 예로 city속성을 위한 데이타는 10%의 키 평균 분포를 가져야 한다. 이러한 제약조건을 만족시키는 데이타의 생성은 여간 번거로운 작업이 아니므로 우리는 자동화 도구를 개발하였으며, 이를 이용하여 데이타를 생성하였다.

각 클래스는 char(20)의 name 속성을 가지고 있다. name 속성을 위한 데이타는 무작위 수 생성기에 의해 생성된 임의의 문자열로 구성되어 있다. 우리는 아스키 파일내의 데이타를 name 속성에 대해 정렬하였다. 정렬을 위해서는 유닉스의 sort 명령어가 사용되었다. name 속성에 대해 정렬된 데이타는 주 키 속성에 대해서 무작위로 섞이는 효과를 가져오며 이렇게 섞인 데이타는 X와 Y에 적재되었다.

우리는 시험 데이타베이스에 대해 각 DBMS에서 지원하는 인덱스(index)나 통계 정보 기능을 충분히 사용하여 최적의 성능이 나오도록 배려하였다. 인덱스는 데이타베이스내의 모든 속성과 where 절에서 언급된 속성들의 조합으로 생성되었다. 우리는 X와 Y에서 지원하는 기능의 정도에 따라 집합형, 사용자 정의 메소드, ADT에 대해서도 인덱스를 생성하였다. 클러스터(cluster) 인덱스와 클래스 계층 인덱스가 가능한 경우에도 이를 사용하여 범위 질의(range query)와 클래스 계층 구조에 대한 검색이 유리하도록 하였다.

성능 평가의 확장 요소는 1로 측정되었고 응답 시간의 측정은 유닉스에서 제공하는 gettimeofday 함수를 사용하였다. 또한 우리는 콜드 런의 효과를 위하여 각 질의 반복 수행 사이에 큰 파일을 메모리로 읽어 버퍼에 캐쉬(cache)된 데이타를 플러시(flush) 하였다. 대부분의 질의에 대하여 이러한 방식의 플러시 효과가 잘 작동하였으나, 그렇지 않은 경우에는 시스템 재부팅(rebooting)을 통하여 버퍼를 비웠다.

우리는 각 시험 질의에 대하여 질의 실행 정확성을 확인하는 질의 실행 검증을 하였다. 각 질의에 대한 질의 실행 검증은 데이타베이스 인스턴스 분포 지식에 기반하여, 예상되는 질의 결과 인스턴스의 개수와 질의 수행 후 실제 반환되는 인스턴스 개수를 비교하였

다. 질의 수행 후 반환되는 인스턴스 개수를 예측하는 것은 우리가 본 벤치마크에서 합성된 데이타베이스만을 사용함으로써 가능할 수 있었다.

최종 성능 평가 결과는 부록 1에 첨부되어 있다. 표에 있는 숫자는 단위가 초이며, 'X' 기호는 해당 시스템이 관련 기능을 제공하지 않는다는 의미이다. 구현의 편이성을 위하여 각 질의 당 최대 수행시간을 2시간을 하였으며, 2시간을 초과하는 질의는 질의 종료 시까지 질의 시간을 측정하지 않았다.

5. 결 론

본 논문에서 우리는 객체-관계형 데이타베이스에 대한 새로운 벤치마크를 제안하였다. 본 벤치마크의 설계 전략, 데이타베이스 스키마, 데이타베이스 인스턴스, 시험 질의를 기술하였다. 본 벤치마크는 객체-관계형 데이타베이스 시스템을 위해 설계되었으므로 객체-관계형 데이타베이스 시스템의 전형적인 기능을 시험하고 있으며, 본 벤치마크의 특징으로는 시험 데이타베이스로서 합성된 데이타베이스만의 사용, 시험 데이타베이스의 확장성, 질의 기반 시험 등이 있다.

우리는 본 벤치마크를 두 개의 상용 객체-관계형 DBMS을 사용하여 구현하였으며, 그 결과를 보고하였다. 불행하게도 우리가 채택한 2개의 객체-관계형 데이타베이스 시스템으로는 DBMS의 기능 부족으로 인하여 본 벤치마크의 모든 사양을 현재로는 구현할 수 없었다. 그러나 우리가 본 벤치마크를 처음 설계할 당시에는 본 벤치마크의 거의 모든 기능을 제공하는 객체-관계형 데이타베이스 시스템이 시장에 존재하고 있었음을 명시하고 싶다. 그 DBMS는 회사 사정으로 인하여 현재 시장에서 사라졌으며 우리는 그 제품의 라이센스(license)를 더 이상 연장할 수 없었다. 우리는 가까운 미래에 시장에서 본 벤치마크의 사양에서 제시한 질의의 대부분을 지원하는 DBMS가 나타나리라 믿는다.

이미지(Images), 사운드(sounds), 비디오(video)와 같은 멀티미디어(multimedia) 데이타는 오늘날의 DBMS에서 흔히 지원된다. 또한 웹의 발달로 인하여 전역 문헌 검색(FTR, full text retrieval)의 필요성이 증대되고 사용자는 DBMS에서 이들 비 구조적 문헌에 대한 검색 능력을 요구한다. 이들 기능은 엄밀히 객체-관계형 데이타베이스의 영역에 있지 않으나 많은 DBMS가 이를 지원한다. 멀티미디어 데이타와 전역 문헌 검색에 대한 벤치마크는 향후 과제로 남긴다.

참 고 문 헌

[1] M. Asgarian, M. J. Carey, D. J. DeWitt, J. Gehrke, J. F. Naughton, and D. N. Shah, "The BUCKY Object-Relational Benchmark," Proc. of the ACM SIGMOD Conference, 1997.

[2] D. Bitton and C. Turbyfill, "A Retrospective on the Wisconsin Benchmark," In Readings in Database Systems, M. Stonebraker ed., Morgan Kaufmann, 1988.

[3] M. J. Carey, D. J. DeWitt and J. F. Naughton, "The OO7 Benchmark," Proc. ACM SIGMOD Conference, pp.12-21, 1993.

[4] J. Gray, 'The Benchmark Handbook', 2nd Ed., Morgan Kaufmann, 1993.

[5] S. H. Lee, S. J. Kim, W. Kim, "The BORD Benchmark for Object-relational Databases," Submitted for publication, 2000.

[6] M. Stonebraker, J. Frew, K. Gardels and J. Meredith, "The SEQUOIA 2000 Storage Benchmark," Proc. of the ACM SIGMOD Conference, pp.2-11, 1993.

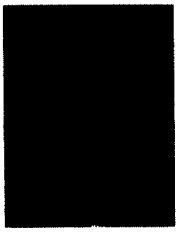
[7] M. Stonebraker, 'Object-relational DBMSs', Morgan Kaufmann, 1996.

[8] C. Turbyfill, C. Orji and D. Bitton, "AS3AP : An ANSI SQL Standard Scaleable and Portable Benchmark for Relational Database Systems," In The Benchmark Handbook, pp.317-358, J. Gray ed., Morgan Kaufmann, 1993.

[9] W. Kim, 'Completeness Criteria for Object-Relational Database Systems', UniSQL Inc., 1996.

부 록 질의 응답시간 결과

분 류	질의번호	X	Y
클래스	질의 1-1	1.6	2.6
	질의 1-2	1.9	45.1
객체 아이디	질의 2-1	X	2.7
	질의 2-2	X	2.7
조 인	질의 3-1	110.3	748.0
	질의 3-2	4.6	6.2
	질의 3-3	23.7	33.9
참 조	질의 4-1	X	6.7
	질의 4-2	X	31.9
	질의 4-3	X	7.7
	질의 4-4	X	38.3
집 합	질의 5-1	X	99.0
	질의 5-2	X	136.9
	질의 5-3	X	2598.0
	질의 5-4	X	2763.7
메소드	질의 6-1	57.8	101.8
	질의 6-2	56.8	50.9
	질의 6-3	116.4	Memory Leak
연산자	질의 7-1	X	X
	질의 7-2	X	X
	질의 7-3	X	X
플래트닝	질의 8-1	X	X
ADT 접근	질의 9-1	10.5	X
	질의 9-2	6.6	15.8
ADT 연산	질의 10-1	10.5	X
	질의 10-2	2.8	38.7
ADT 조인	질의 11-1	2시간 이상 소요	X
	질의 11-2	2시간 이상 소요	X



김 성 진

e-mail : 1ace@nownuri.net

1998년 숭실대학교 소프트웨어 공학과 졸업(학사)

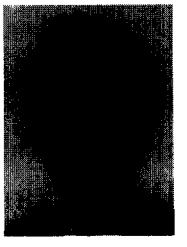
2000년 숭실대학교 대학원 컴퓨터학과(석사)

2000년~현재 숭실대학교 컴퓨터 학과 대학원 박사과정

2000년~현재 가천 길 대학 강사

2000년~현재 숭실대학교 전산원 강사

관심분야 : 인터넷 데이타베이스, 데이타베이스 시스템 성능평가



이 상 호

e-mail : shlee@computing.soongsil.ac.kr

1984년 서울대학교 전산공학과 졸업(학사)

1986년 미국 노스웨스턴대 전산학과(석사)

1989년 미국 노스웨스턴대 전산학과(박사)

1990년~1992년 한국전자통신 연구원, 선임연구원

1999년~2000년 미국 조지 메이슨대 소프트웨어 정보공학과 교환 교수

1999년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가, WWW